Algebra Booleana

Corso di Architettura degli elaboratori e laboratorio

Modulo Laboratorio

Gabriella Verga

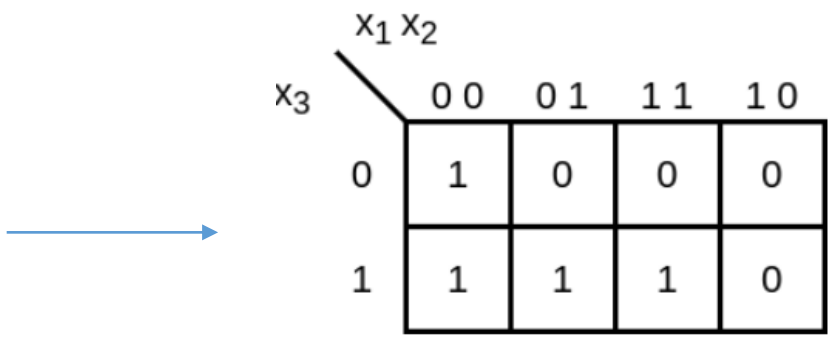
Mappe di Karnaugh

Mappe di Karnaugh

Le Mappe di Karnaugh sono un metodo di tipo geometrico che permettono di ricavare rapidamente l'espressione logica di **costo minimo** della funzione. L'essenza del metodo è di rappresentare la tabella di verità in forma differente.

Vantaggioso per funzioni con poche variabili (3 o 4).

x_1	x_2	x_3	f_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Mappe di Karnaugh

Sono ordinate in modo che caselle adiacenti abbiano solo una variabile dal valore differente.

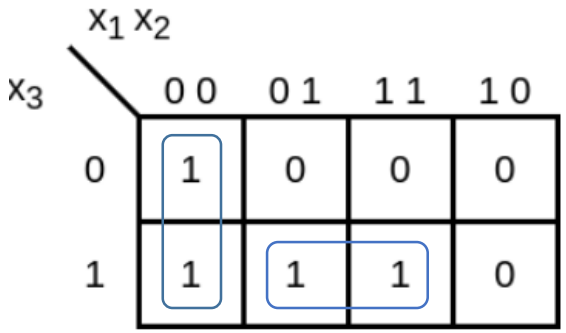
Il principio delle mappe di Karnaugh e che : due quadrati adiacenti differiscono nel valore di una sola variabile.

Se accade che due caselle contengono il valore 1 e sono adiacenti nel senso geometrico (hanno un lato in comune) c'è la possibilità di effettuare un passo di semplificazione della funzione.

x_3 \ $x_1 x_2$	0 0	0 1	1 1	1 0
0	1	0	0	0
1	1	1	1	0

Come si usano

1. Raggruppare le caselle di valore 1 adiacenti orizzontalmente o verticalmente. Ogni gruppo deve presentare un numero di caselle contenenti un 1 pari a potenza di 2 (1,2,4,8).
2. Ogni gruppo rappresenta il prodotto delle sue variabili con lo stesso valore (forma **diretta** se **1** e **negata** se **0**). Identificare quali variabili non contribuiscono ed eliminarle nella forma analitica.
3. Somma dei gruppi creati.



Box blu: x_3 non contribuisce → $\overline{x_1}\,\overline{x_2}$
Box blu: x_3 non contribuisce → $x_2\,x_3$
$f_1 = \overline{x_1}\,\overline{x_2} + x_2\,x_3$

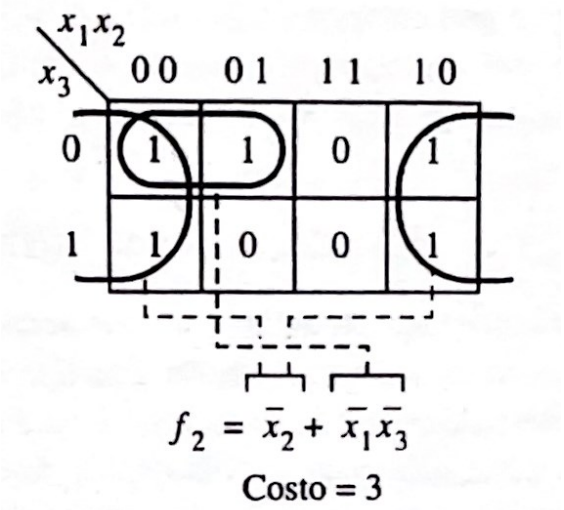
Altri esempi

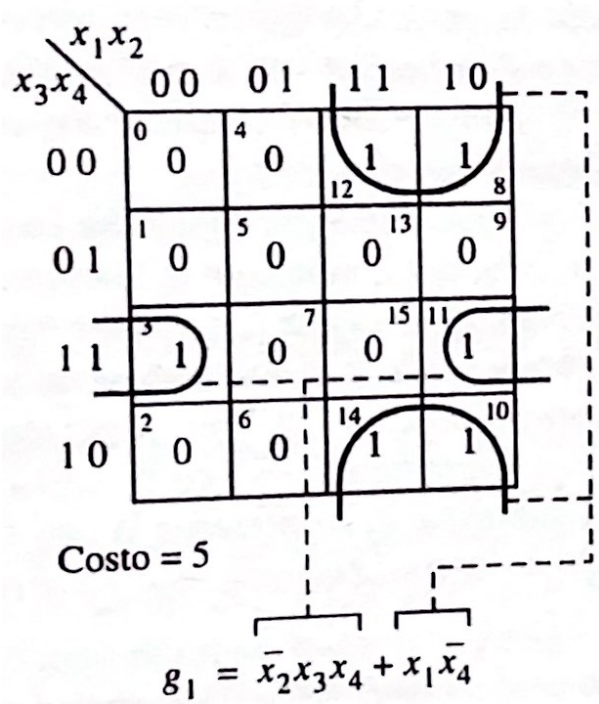
x_3 \ x_1x_2	00	01	11	10
0	1	1	0	1
1	1	0	0	1

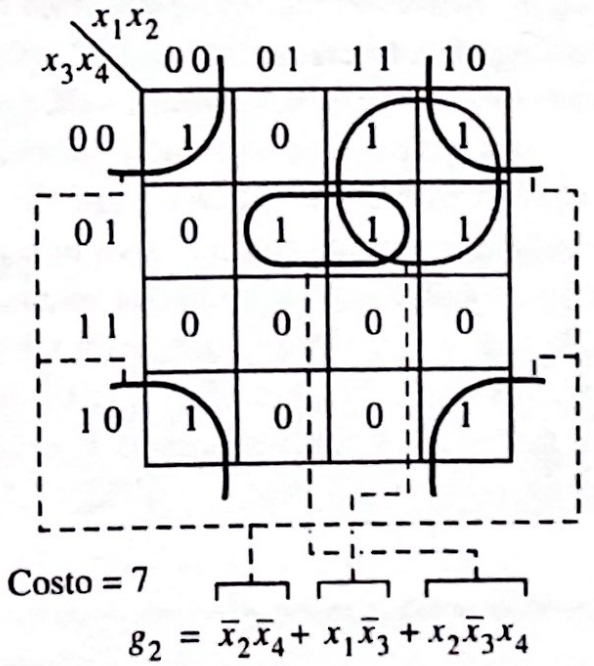
x_3x_4 \ x_1x_2	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	0	0	1
10	0	0	1	1

x_3x_4 \ x_1x_2	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	0	0	0
10	1	0	0	1

Soluzioni


x_1x_2
x_3
00 01 11 10
0 1 1 0 1
1 1 0 0 1
$f_2 = \bar{x}_2 + \bar{x}_1\bar{x}_3$
Costo = 3


x_1x_2
x_3x_4
00 01 11 10
0 4 12 8
0 0 1 1
1 5 13 9
0 0 0 0
3 7 15 11
1 0 0 1
2 6 14 10
0 0 1 1
Costo = 5
$g_1 = \bar{x}_2x_3x_4 + x_1\bar{x}_4$


x_1x_2
x_3x_4
00 01 11 10
00 1 0 1 1
01 0 1 1 1
11 0 0 0 0
10 1 0 0 1
Costo = 7
$g_2 = \bar{x}_2\bar{x}_4 + x_1\bar{x}_3 + x_2\bar{x}_3x_4$

Condizione di indifferenza

Condizione di indifferenza

- Spesso capita che una funzione logica non sia definita su tutte le combinazioni di valori delle sue variabili
- Le variabili non usate si dice siano in condizione di indifferenza (don’t care condition)
- Nella tabella di verità vengono indicate con il simbolo “X”
- Il loro valore (0 o 1) si può scegliere in modo arbitrario. Naturalmente conviene fare scelte che conducano alla semplificazione più spinta, ovvero a costo minimo.

Come si usa

Cifra decimale	#	Codifica binaria b_3	b_2	b_1	b_0	f
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	1
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	1
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	1
Non usate	10	1	0	1	0	x
	11	1	0	1	1	x
	12	1	1	0	0	x
	13	1	1	0	1	x
	14	1	1	1	0	x
	15	1	1	1	1	x

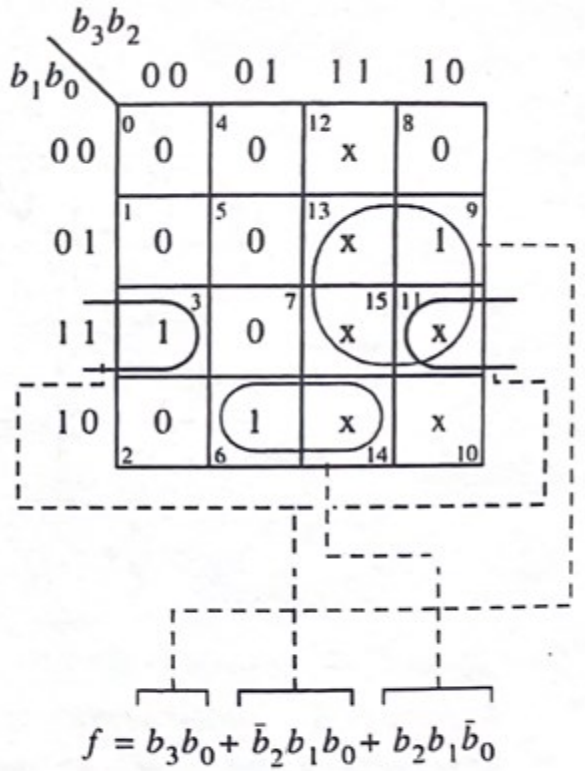
(a) Tabella di verità

b_1b_0 \ b_3b_2	00	01	11	10
00	0	0	x	0
01	0	0	x	1
11	1	0	x	x
10	0	1	x	x

Come si usa

Cifra decimale	#	Codifica binaria b_3	b_2	b_1	b_0	f
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	1
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	1
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	1
Non usate	10	1	0	1	0	x
	11	1	0	1	1	x
	12	1	1	0	0	x
	13	1	1	0	1	x
	14	1	1	1	0	x
	15	1	1	1	1	x

(a) Tabella di verità

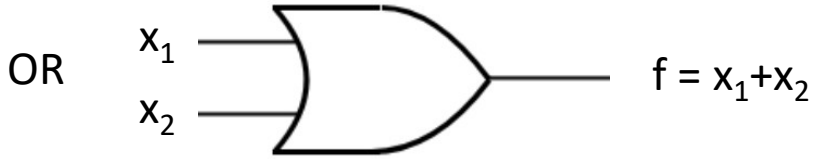


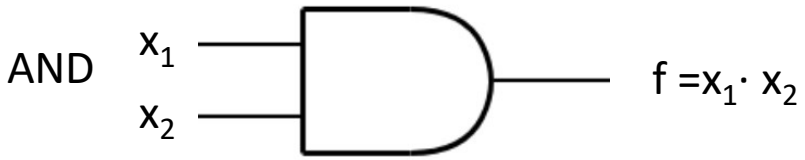
(b) Mappa a quattro variabili

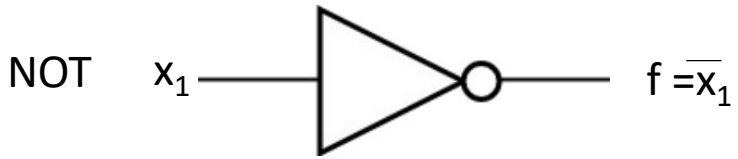
Circuiti Logici

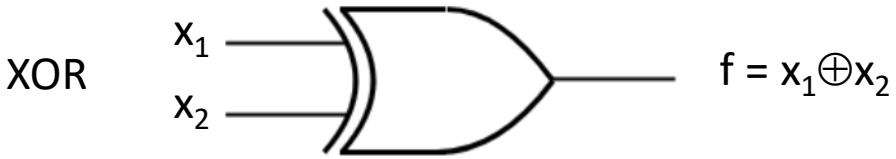
Circuiti Logici

- Le operazioni logiche (AND, OR, NOT, XOR) possono essere realizzate da semplici circuiti elettronici. **Questi circuiti base vengono chiamati PORTE.**







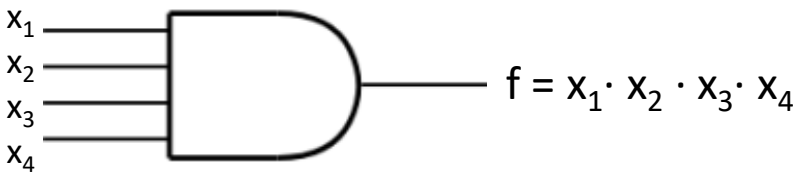


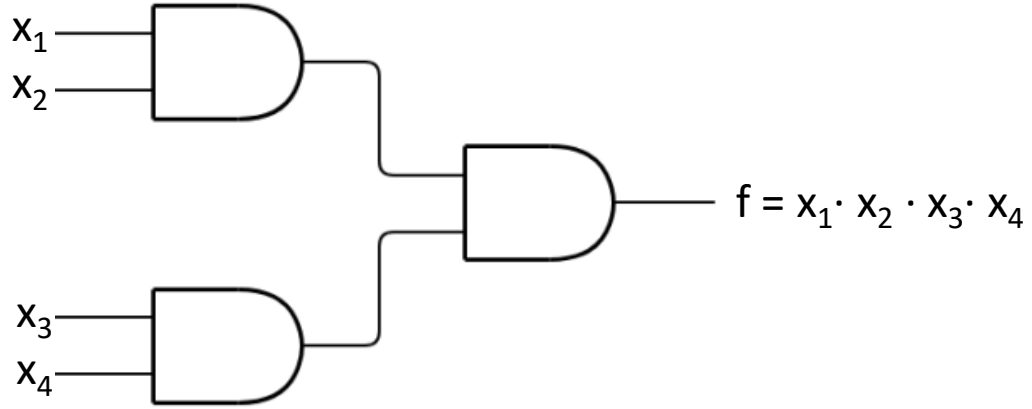
Collegando entrate e uscite delle porte si possono rappresentare le reti combinatorie, creando una rete di porte logiche collegate tra loro. Una rete combinatoria ha n ingressi binari ed m uscite binarie con n e m ≥ 1.

Porte a più ingressi

- Grazie alla **proprietà associativa** AND e OR possono essere estese a più di 2 ingressi, ovvero mettere in due livelli a cascata o ad albero porte AND o OR a due ingressi.
- ESEMPIO:

$$\mathbf{f = x_1 \cdot x_2 \cdot x_3 \cdot x_4} \rightarrow \textbf{porta AND con 4 ingressi}$$

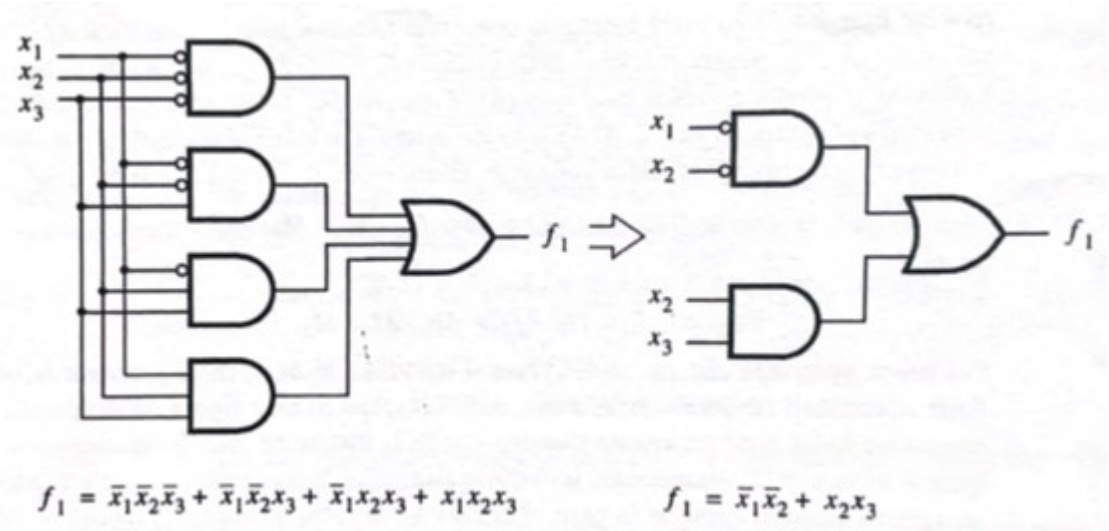




Reti combinatorie

- E’ possibile rappresentare un’espressione logica come rete combinatoria con:
  - una porta per ogni operatore logico
  - collegando le porte tra loro ad albero seguendo i livelli di priorità nell’espressione

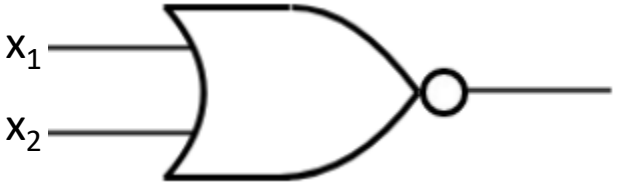
Le espressioni *SOP* e POS corrispondono a reti a due livelli.



NAND e NOR

- Sono equivalenti alle funzioni AND e OR seguite da porta NOT (rispettivamente).
- Si denota tramite gli operatori a due argomenti "↑" o "↓" rispettivamente.
- NAND e NOR sono porte **UNIVERSALI**, ovvero si può realizzare una qualsiasi funzione combinatoria con reti logiche di soli NAND o soli NOR.

NOR



$f = x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$

NAND

$f = x_1 \uparrow x_2 = \overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$

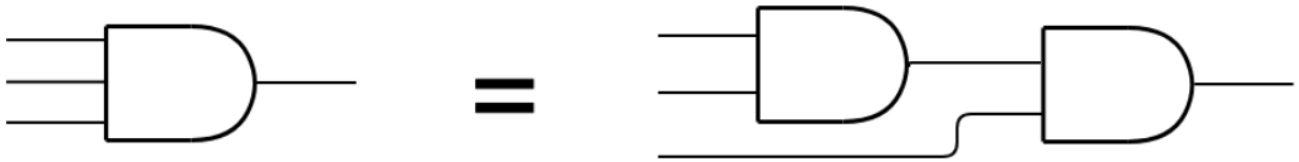
x_1	x_2	$\overline{x_1+x_2}$	$\overline{x_1x_2}$
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

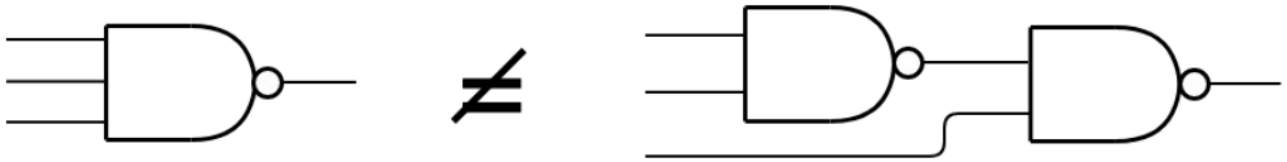
Proprietà

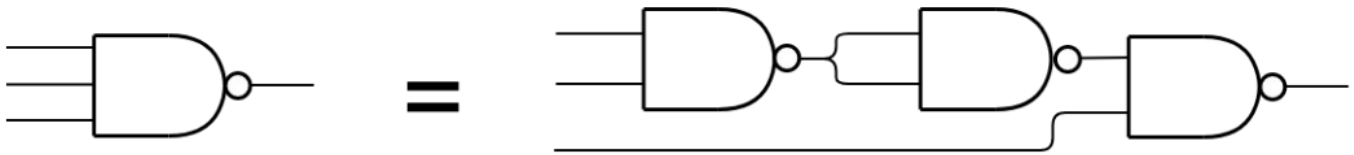
- NAND e NOR
- Godono della proprietà commutativa
- **NON** godono della proprietà associativa

Proprietà associativa

- Dato che NAND e NOR **NON** godono della proprietà associativa NON è possibile scomporlo come albero o cascata di porte.







Trasformazione di un'espressione (SOP)

- Grazie alle leggi di De Morgan e alla legge di involuzione possiamo passare da una **SOP** ad una rete di **NAND**:

$$
\begin{aligned}
(x_1 \uparrow x_2) \uparrow (x_3 \uparrow x_4) &= \overline{(\overline{x_1 \cdot x_2}) \cdot (\overline{x_3 \cdot x_4})} = \quad (De\ Morgan) \\
&= \overline{\overline{x_1 x_2}} + \overline{\overline{x_3 x_4}} = \quad (Involuzione) \\
&= x_1 x_2 + x_3 x_4 \quad (Sum\ of\ Products)
\end{aligned}
$$